History of C

The story of C began as a consequence of the advent of Unix, which was developed in 1969 by Ken Thompson. He was a computer programmer at AT& T Bell Laboratories who had been using an older multiuser operating system called Multics. The first Unix was written in PDP-7 assembly language to run on a DEC PDP-7 machine. Soon afterwards a compiler for a new language called B was implemented, and with a new computer DEC PDP-11 plans were made to rewrite Unix in B. B was influenced by another older language BCPL. But B turned out to be unsuitable for the new machine, and so another language was developed in place of it for the implementation of Unix. That language was C.

The alphabet

An alphabet is a set of characters or letters. Since C was developed by Dennis Ritchie during the 1970s at AT& T Bell Laboratories in the United States, it is not surprising that the alphabet used for the purpose is ASCII (American National Standard Code for Information Interchange).

Decimal	*Octal*	*Hexadecimal*	*Character*
0	0	0	(null, NUL)
1	1	1	(SOH)
2	2	2	(STX)
3	3	3	(ETX)
4	4	4	(EOT)
5	5	5	(ENQ)
6	6	6	(ACK)
7	7	7	(bell, BEL)
8	10	8	(backspace, BS)
9	11	9	(horizontal tab, HT)
10	12	a	(newline, line feed, LF)
11	13	b	(vertical tab, VT)
12	14	c	(form feed, FF)
13	15	d	(carriage return, CR)
14	16	e	(SO)
15	17	f	(SI)
16	20	10	(DLE)
17	21	11	(DC1)
18	22	12	(DC2)
19	23	13	(DC3)
20	24	14	(DC4)
21	25	15	(NAK)
22	26	16	(SYN)
23	27	17	(ETB)
24	30	18	(CAN)
25	31	19	(EM)
26	32	1a	(SUB)
27	33	1b	(ESC)
28	34	1c	(FS)
29	35	1d	(GS)
30	36	1e	(RS)
31	37	1f	(US)
32	40	20	(blank)

Table 1 *ASCII characters*

Decimal	*Octal*	*Hexadecimal*	*Character*
33	41	21	!
34	42	22	"
35	43	23	#
36	44	24	$
37	45	25	%
38	46	26	&
39	47	27	'
40	50	28	(
41	51	29	)
42	52	2a	*
43	53	2b	+
44	54	2c	,
45	55	2d	-
46	56	2e	.
47	57	2f	/
48	60	30	0
49	61	31	1
50	62	32	2
51	63	33	3
52	64	34	4
53	65	35	5
54	66	36	6
55	67	37	7
56	70	38	8
57	71	39	9
58	72	3a	:
59	73	3b	;
60	74	3c	<
61	75	3d	=
62	76	3e	>
63	77	3f	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4a	J
75	113	4b	K
76	114	4c	L
77	115	4d	M
78	116	4e	N
79	117	4f	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S

Table 1 *ASCII characters (continued)*

Decimal	*Octal*	*Hexadecimal*	*Character*
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5a	Z
91	133	5b	[
92	134	5c	\
93	135	5d	]
94	136	5e	^
95	137	5f	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6a	j
107	153	6b	k
108	154	6c	l
109	155	6d	m
110	156	6e	n
111	157	6f	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7a	z
123	173	7b	{
124	174	7c	|
125	175	7d	}
126	176	7e	~
127	177	7f	(DEL)

Table 1 *ASCII characters (continued)*

Table 1 gives ASCII alphabet, the whole set of its characters. The letters from 0 to 32 as well as 127 are control characters, and are described by their function in parenthesis. Example 1 is the first programme according to the universal programmer's lore.

Example 1.

```
1 /* the first programme, Kit Tyabandha, 20 Nov 06 */
2 #include<stdio.h>
3 int main(void){
4   printf("\n Hello World!\n\n"); return(0);
5 }
```

The output of this is simply this.

```
 Hello World!
```

Example 2 shows a simple programme that reports the decimal code in ASCII for a given character. Example 3 gives the codes that list ASCII characters together with their corresponding decimal, octal and hexadecimal representations.

Example 2.

```
1 /* input and output a letter, Kit Tyabandha, 19 Nov 06 */
2 #include<stdio.h>
3 main(){
4   char *in;
5   printf("Input a character: ");
6   in =getchar();
7   printf("The character is %d\n\n", in);
8   return(0);
9 }
```

A sample run of this programme gives the following output.

```
Input a character: c
The character is 99
```

Example 3.

```
1 /* lists ASCII together with its decimal, octal and hexadecimal codes
2                   Kit Tyabandha, 20 Nov 06 */
3 #include<stdio.h>
4 main(){
5   int i;
6   printf("ASCII\n");
7   printf("Decimal\tOctal\tHexadecimal\tCharacter\n\n");
8   for(i=0; i<=127; i++){
9     printf(" %d\t %o\t %x\t\t %c\n",i,i,i,i);
10  }
11  return(0);
12 }
```

Data types

C groups data into various types. Table 2 gives a list of these together with their ranges.

Type	*Signedness*	*Size(bytes)*	*Range*
char	signed	1	-128–127
int	signed	2	-32768–32768
float	signed	4	$3.4e \pm 38$(7-digit accuracy)
double	signed	8	$1.7e \pm 308$(15-digit accuracy)
void	neither	0	no values

Operators

There are two kinds of operator, that is *relational* and *logical*. Table's 3 and 4 show respectively these.

Operator	*Function*
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
==	equal to
! =	not equal to

Table 3 *Relational operators*

Operator	*Function*
&&	AND
||	OR
!	NOT

Table 4 *Logical operators*

Library functions

Often called *commands* they are in fact *functions* which are defined in one of those numerous *headers*. These latter are sets of *macros*. The beauty of C is that one could write a new header file for his own use.

`printf`

The formatted print command, `printf`, uses a number of format specifiers and escape sequences, which are shown in Table's5 and respectively 6. The specifiers are also called *specifications*. Those other than %n and %% are also known as *conversion characters*.

Specifier	*Representation*
%c	character
%d	decimal integer
%e	scientific notation
%E	scientific notation
%f	floating point
%g	either %e or %f, the shorter one
%G	either %E or %F, the shorter one
%i	integer, the same as %d
%n	integer pointer holding number of characters printed
%o	octal integer
%p	pointer
%s	string
%u	unsigned decimal integer
%x	hexadecimal integer, in lower case
%X	hexadecimal integer, in upper case
%%	per cent sign

Table 5 *Format specifications for* `printf`

Escape	*Character*
\a	bell (alert)
\b	backspace
\t	horizontal tab
\n	newline, line feed
\v	vertical tab
\f	form feed
\r	carriage return
\"	quotation mark
\'	apostrophe
\?	question mark
\\	backslash
\0	null
\ooo	octal digits
\xhh	hexadecimal digits

Table 6 *Escape sequences for* `printf`

scanf

The formatted scan, `scanf`, has as its conversion characters those listed in Table 7. Example 4 gives a preliminary study of how the function work.

Conversion	*Description*
c	single character
d	decimal integer
e	floating point
f	floating point
g	floating point
h	short integer
i	decimal, octal, or hexadecimal integer
o	octal integer
s	string
u	unsigned decimal integer
x	

Table 7 *Conversion characters for* `scanf`

Example 4.

```
1 /* scanf study, Kit Tyabandha, 21 Nov 06 */
2 int main(){
3   char in;
4   printf("\nEnter a character: ");
5   scanf("%c", &in);
6   printf("The character is %c\n\n", in);
7   return(0);
8 }
```

Processing texts

Example 5 shows how text output can be formatted as to appear on screen.

Example 5.

```
1 main(){
2   printf("\n\t\t A Prayer (I)\n\n\
3    Of a private but not privative life,\n\
4    possibly deprived but never depraved,\n\
5    objective even if objected never objectionable,\n\
6    reading never leading,\n\
7    surrendering never surprising,\n\n\
8    \t\t\t Kit Tyabandha, (c)2005\n");
9   return(0);
10 }
```

Bibliography

Byron Gottfried. *Programming with C.* Schaum's Outlines Series, McGraw-Hill, 1996

Steven Holzner. *C Programming.* Brady, 1991

Robert C Hutchison and Steven B Just. *Programming using the C language.* Computer Science Series, McGraw-Hill,